SETTINGS AND CONSTRAINTS VALIDATION TO ENABLE DESIGN FOR OPERATIONS

FIELD OF THE INVENTION

[01] Aspects of the present invention relate to software development for distributed computing systems. More specifically, aspects of the present invention provide software tools that allow developers to conveniently identify and satisfy constraints that apply to applications and hosting environments.

BACKGROUND

[02] Distributed computing systems typically include applications that are hosted on a plurality of computer devices. One type of distributed computer system is a data center (such as an Internet data center (IDC) or an Enterprise Data Center (EDC)), which is a specifically designed complex that houses many computers for hosting network-based services. Data centers, which may also go by the names of "Webfarms" or "server farms", typically house hundreds to thousands of computer devices that form a hosting environment and are located in climate-controlled, physically secure buildings. Each of the computer devices has operational requirements that must be met by applications hosted on the computer device. Similarly, each application has operational requirements that must be met by the hosting environment.

[03] Applications that will be hosted on distributed computing systems are typically configured to operate with a single computer device during the development phase. For example, all of the settings associated with the application are set to meet the operational requirements of a single server. During the deployment of the application, extensive modifications to the application and hosting environment settings are often required to ensure that the application settings meet the operational requirements of the hosting

environment and that the hosting environment settings meet the operational requirements of the application. This process can be time consuming and expensive.

[04] Therefore, there is a need in the art for design tools and methods that facilitate identifying and satisfying hosting environment and application operational requirements during the design phase of an application.

BRIEF SUMMARY

[05] Aspects of the present invention address one or more of the issues mentioned above, thereby providing design tools and methods that allow developers to identify and satisfy application and hosting environment constraints. An application is modeled such that the model includes the identification of constraints that are placed on a hosting environment that hosts the application. The hosting environment is also modeled such that the model identifies constraints that are placed on the application. During a validation stage, a design tool determines whether the constraints have been satisfied. Identifying and satisfying application and hosting environment constraints during the development phase reduces deployment modifications and the time required to deploy applications.

BRIEF DESCRIPTION OF THE DRAWINGS

[06] Aspects of the present invention are described with respect to the accompanying figures, in which like reference numerals identify like elements, and in which:

[07] Figure 1 shows a functional block diagram of a conventional general-purpose computer system;

[08] Figure 2 shows a system and method for setting and validating application and hosting environment constraints, in accordance with an embodiment of the invention;

[09] Figure 3 shows a design surface that may be used to develop an application, in accordance with an embodiment of the invention;

[10] Figure 4 illustrates a design surface that may be used to develop a hosting environment, in accordance with an embodiment of the invention;

[11] Figure 5 illustrates a design surface that may be used to bind an application to a hosting environment, in accordance with an embodiment of the invention; and

[12] Figure 6 illustrates an alternate design surface for binding applications to hosting environments.

DETAILED DESCRIPTION

Exemplary Operating Environment

[13] Figure 1 is a functional block diagram of an example of a conventional general-purpose digital computing environment that can be used to host design tools that implement various aspects of the present invention. In Figure 1, a computer 100 includes a processing unit 110, a system memory 120, and a system bus 130 that couples various system components including the system memory to the processing unit 110. The system bus 130 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory 120 includes read only memory (ROM) 140 and random access memory (RAM) 150.

[14] A basic input/output system 160 (BIOS), containing the basic routines that help to transfer information between elements within the computer 100, such as during start-up, is stored in the ROM 140. The computer 100 also includes a hard disk drive 170 for reading from and writing to a hard disk (not shown), a magnetic disk drive 180 for reading from or writing to a removable magnetic disk 190, and an optical disk drive 191

for reading from or writing to a removable optical disk 192 such as a CD ROM or other optical media. The hard disk drive 170, magnetic disk drive 180, and optical disk drive 191 are connected to the system bus 130 by a hard disk drive interface 192, a magnetic disk drive interface 193, and an optical disk drive interface 194, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the personal computer 100. It will be appreciated by those skilled in the art that other types of computer readable media that can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the example operating environment.

[15] A number of program modules can be stored on the hard disk drive 170, magnetic disk 190, optical disk 192, ROM 140 or RAM 150, including an operating system 195, one or more application programs 196, other program modules 197, and program data 198. A user can enter commands and information into the computer 100 through input devices such as a keyboard 101 and pointing device 102. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner or the like. These and other input devices are often connected to the processing unit 110 through a serial port interface 106 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). Further still, these devices may be coupled directly to the system bus 130 via an appropriate interface (not shown). A monitor 107 or other type of display device is also connected to the system bus 130 via an interface, such as a video adapter 108. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

[16] The computer 100 can operate in a networked environment using connections to one or more remote computers, such as a remote computer 109. The remote computer 109 can be a server, a router, a network PC, a peer device or another common network node, and typically includes many or all of the elements described above relative to the computer 100, although only a memory storage device 111 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 112 and a wide area network (WAN) 113. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[17] When used in a LAN networking environment, the computer 100 is connected to the local network 112 through a network interface or adapter 114. When used in a WAN networking environment, the personal computer 100 typically includes a modem 115 or other means for establishing communications over the wide area network 113, such as the Internet. The modem 115, which may be internal or external, is connected to the system bus 130 via the serial port interface 106. In a networked environment, program modules depicted relative to the personal computer 100, or portions thereof, may be stored in the remote memory storage device.

[18] It will be appreciated that the network connections shown are illustrative and other techniques for establishing a communications link between the computers can be used. The existence of any of the various well-known protocols such as TCP/IP, Ethernet, FTP, HTTP, Bluetooth, IEEE 802.11x and the like is presumed, and the system can be operated in a client-server configuration to permit a user to retrieve web pages from a web-based server. Any of various conventional web browsers can be used to display and manipulate data on web pages.

Description of Illustrative Embodiments

[19] Figure 2 illustrates a system and method for setting and validating application and hosting environment constraints. A development tool 200 is an application that is used to design a system. An exemplary design tool is Microsoft® Visual Studio® development system. Development tool 200 includes an application designer module 202 that provides a design surface for designing an application. A logical infrastructure designer module 204 provides a design surface for designing a hosting environment. A system deployment diagram module 206 binds the application to the hosting environment and generates a user interface that allows a developer to correct configuration errors.

[20] Abstract type model documents 208 may include frameworks that list settings for applications and hosting environments. For example, a server may have several settings that may be set, such as whether the server only hosts web applications that use SOAP, require secure sockets, provide script access, utilize a minimum bandwidth, provide load balancing, include a minimum amount of memory, etc. Application settings may relate authentication modes and protocols, whether secure SSL is required, whether impersonating is allowed, etc.

[21] When a developer selects elements when designing a system, in step 201, the abstract type model documents 208 for those elements are accessed by development tool 200. Figure 3 shows a design surface 300 that may be used to develop an application, in accordance with an embodiment of the invention. Design surface 300 may be created with Application designer module 202. A developer may select elements from column 302 and drag them into region 304. Constraints that the application will impose on the hosting environment may be listed in section 306. Constraints section 306, for example, indicates that the application is imposing the constraint of requiring the hosting environment to be an Internet Information Services (IIS) 6.0 host. Application settings

may be set by making appropriate selections in sections 308 and 310. The settings options for applications may be obtained from abstract type model documents 208.

[22] Figure 4 illustrates a design surface 400 that may be used to develop a hosting environment, in accordance with an embodiment of the invention. Elements may be selected from column 402 and draged into region 404. Constraints that the hosting environment will impose on the application may be listed in sections 406 and 408. Constraints section 406, for example, indicates that the hosting environment is imposing the constraint of requiring the application to use ASP.NET security. Application settings may be set by making appropriate selection in section 410. There can be any number of constraints that, when selected, in section 406 will cause a new section to replace section 408. The setting options for hosting environments may be obtained from abstract type model documents 208.

[23] A settings and constraints editor 210 may be invoked in step 203 to allow a developer to select settings and define constraints. In an alternative embodiment, settings and constraints editor 210 may be part of development tool 200.

[24] In step 204, settings and constraints editor 210 receives settings and constraints data from development tool and operates on concrete type model documents 212 in step 205. Concrete type model documents 212 may be similar abstract type model documents 208 but include specific settings and constraints.

[25] Figure 5 illustrates a design surface 500 that may be used to bind an application to a hosting environment. Application modules included in section 502 may be selected and dragged to the hosting environment elements shown in section 504. In one embodiment of the invention, design tool 200 creates concrete type model documents that use the System Definition Model (SDM) to model applications and hosting environments in step 207. After an application is bound to a hosting environment, concrete type model

documents are passed to a validation module 214 in step 209. Validation module 214 may be accessed by, or be part of, development tool 200.

[26] Validation module 214 compares the models of the application and hosting environment included in concrete type model documents 212 to determine whether application settings satisfy hosting environment constraints and/or hosting environment settings satisfy application constraints. Validation results may then be passed back to development tool 200.

[27] Section 506 of design surface 500 (shown in Figure 5) lists validation errors. In one embodiment of the invention, some or all of the errors listed in section 506 include links to the user interface elements that failed the validation step. For example, if an error listed in section 506 relates to a configuration error for web application 1, selecting a link may generate section 304 (shown in Figure 3) with element 312 highlighted or marked. In one embodiment error icons may be displayed next to the relevant elements.

[28] Figure 6 illustrates an alternate design surface for binding applications to hosting environments. The application elements shown in section 602 may be bound to the hosting elements shown in section 604 by dragging the application elements to the hosting elements. Section 606 may list validation errors that are generated in the manner described above.

[29] The present invention has been described in terms of preferred and exemplary embodiments thereof. Numerous other embodiments, modifications and variations within the scope and spirit of the appended claims will occur to persons of ordinary skill in the art from a review of this disclosure.